Distributed Temporal Link Prediction Algorithm based on Label Propagation

Xiaolong Xu

School of Computer Science
Nanjing University of Posts and Telecommunications
Nanjing, China

Nan Hu

State Key Laboratory of Information Security
Chinese Academy of Sciences
Beijing, China

Tao Li

Jiangsu Key Laboratory of Big Data Security Intelligent Processing
Nanjing University of Posts and Telecommunications
Nanjing, China

Marcello Trovati and Georgios Kontonatsios

Department of Computer Science
Edge Hill University
Ormskirk, UK

Aniello Castiglione and Francesco Palmieri

Department of Computer Science
University of Salerno
Fisciano (SA), Italy

Abstract

Link prediction has steadily become an important research topic in the area of complex networks. However, the current link prediction algorithms typically neglect the network evolution and tend to exhibit low accuracy and scalability when applied to large-scale organisations. In this article, we propose a novel distributed temporal link prediction algorithm based on label propagation (DTLPLP), governed by the dynamical properties of the interactions between nodes. In particular, nodes are associated with labels, which include details of their sources and the corresponding similarity value. When such labels are

propagated across neighbouring nodes, they are updated based on the weights of the incident links, and the values from same source nodes are aggregated to evaluate the scores of links in the predicted network. Furthermore, DTLPLP has been designed to be distributed and parallelised, and thus is suitable for large-scale network analysis. As part of the validation process, we have designed a prototype system developed in Pregel, which is a distributed network analysis framework. Experiments are conducted on the Enron e-mail network and the General Relativity and Quantum Cosmology Scientific Collaboration network. The experimental results show that when compared to the most of link prediction algorithms, DTLPLP offers enhanced accuracy, stability and scalability.

Keywords: Complex Networks, Network Dynamics, Link Prediction, Label Propagation

1. Introduction

The increasing success and continuous growth of social networks has led to more efficient and faster communication between individuals and to the rapid diffusion of information and knowledge. These organisations can be modelled as complex networks characterised by non-trivial topological properties, experiencing connection dynamics between the composing nodes that can be seen as neither totally regular nor totally random. For example, they may experience assortativity or disassortativity among nodes, or exhibit a scale free behaviour with heavy tail in the degree distribution as well as compliance to power laws, together with noticeable clustering dynamic and the emergence of community structures. Great efforts have been recently devoted to the analysis of these properties and evolution behaviours, in order to better understand how interactions between nodes evolve over time. An important aspect of complex network analysis is *link prediction*, which includes the assessment of potential links and the prediction of future connections [1].

There are many real-world applications for the link prediction technology [2].

For example, indirect relationships between individuals in an online social network system can be discovered in order to build relational knowledge. This can be subsequently used as a "friend" recommendation mechanism to identify the triangular relationships, and thus promoting the adhesive capacity of a social network platform [3]. Another application includes criminal communication networks, which are often investigated to analyse the organisational structure of criminal groups and identify their key figures. For example, in [4], the authors examine the global terrorist data (GTD) based on the social network link analysis and demonstrate the effectiveness of the link prediction technology in mining terrorist relations.

Link prediction technology can also be applied to analyse the correlation between the contents of web pages, and the prediction results can be used to define a knowledge map.

In [5] a Conditional Independent Generalised Relational Topic model (CI-gRTM) is introduced to predict links in multi-modal data (such as multilingual documents and images). In [6] link prediction is utilised to improve the performance of Twitter friend recommendation system based on users' attribute semantics. Furthermore, link prediction is used to determine correlations between current and future diseases that patients may suffer from [7], as well as the exploration of the association between knowledge maps [8]. However, current approaches do not take into consideration the network's temporal evolution information, and do not efficiently scale up to large networks.

In this article, we propose a novel distributed temporal link prediction algorithm based on label propagation (DTLPLP). The main contributions of this work can be summarised as follows:

- Evaluation of network temporal information via network compression techniques to incorporate the frequency of temporal interactions between nodes into the weights of their corresponding links. This is followed by an extension of the label propagation algorithm to effectively improve the accuracy

of the future link prediction. During the process of label propagation, the similarity values of labels are updated by the link weights combined with temporal information. These are subsequently aggregated with suitably defined node keys which form the final link scores, based on the scale of the network or on other specific requirements.

- Large scale complex networks can be efficiently analysed via the distributed and parallelised DTLPLP. In this context, DTLPLP is designed according to the Bulk Synchronous Parallel (BSP) modelling framework, which allows easy implementation of the algorithm on the current mainstream big data processing platforms and has good scalability.

The rest of the article is organised as follows: Section 2 discusses previous approaches to link prediction for complex networks, and Section 3 describes the features characterising the problem addressed in this work. Section 4 introduces the temporal link prediction algorithm and its parallelisation, while Section 5 focuses on the validation process. Finally, Section 6 summarises the main contributions and proposes future research directions.

2. Related Work

Link prediction is an important research area of knowledge discovery in complex networks (refer to [1] for a survey). In particular, Liben-Nowell et al. [2] have proposed one the of the earliest link prediction algorithms focusing on an enhancement of various node similarity indexes. More specifically, Lü [1] divides early link prediction methods into three categories: (1) link prediction based on similarity; (2) link prediction based on maximum likelihood estimation; (3) link prediction based on probabilistic models. Among them, models based on structural similarity are widely used due to their simplicity, low complexity and high prediction performance. Zhou [9] has evaluated the performance of nine common local-information-based similarity indexes on multiple datasets, and has demonstrated that resource allocation (RA) and common neighbours (CN)

achieve a relative improved performance. For specific weighted networks, CN, RA and Adamic-Adar (AA) are discussed in [10], where S_{xy}^{WCN} is introduced, which is defined as:

$$S_{xy}^{WCN} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} w(x,z)^{\alpha} + w(z,y)^{\alpha}, \tag{1}$$

for

- $\Gamma(x) \cap \Gamma(y)$ is the set of common neighbours of the nodes x and y
- $w(x,z)$ and $w(z,y)$ are the weights of the link (x,z) and (z,y), respectively.
- α is the correction factor of the weight.

In the above model, although the utilisation of weight information can improve the overall performance of link prediction algorithms, the dynamics of such networks is not considered. However, this is an important factor as the majority of complex networks exhibit an evolving scale and complexity.
Gao et al. [11] have analysed the topological properties of dynamic networks and have evaluated different algorithms for pattern mining tasks. Deng et al. [12] have discussed the limitations of static link prediction methods. In particular, they have proposed a temporal link prediction method, which uses prediction errors based on static link predictions from previous time windows to refine the prediction process. One disadvantage of this method is that it assumes that relationships between different nodes have an equal weight, which is not always the case. In order to integrate time information with the underling prediction algorithm, Zhao et al. [13] have developed the Time-Difference-Labelled Path (TDLP), which combines time information with the structural features into a unified setting, while proposing a temporal link prediction method based on TDLP. This model utilises logistic regression to calculate the link score between two nodes, based on a specific threshold. However, its precision performance declines with increasing values of such threshold, which demonstrates that the predicted links with higher score are unlikely to appear in the future. In order to address the limitation of the static network in representing information,

Ibrahim and Chen [14] have proposed an algorithm to predict the link variations in dynamic networks by integrating temporal information, community structure, and node centrality in the network. However, the performance of their method depends on fine-tuning a large number of parameters. Thus, over-fitting may occur in the process of parameter adjustment. Rapidly evolving networks produce a large amount of real-time information, which results in higher complexity in terms of number of parameters and their inter-dependencies. Zhao et al. [15] have designed a network sketch algorithm based on MinHash and node-biased sampling techniques. The node similarity indexes used in this algorithm are based on Jaccard, CN and AA. However, they are only evaluated in stand-alone simulation experiments based on non-real time data sets such as DBLP (Digital Bibliography and Library Project). Thus, it remains largely unknown whether such algorithms can be applied to data streaming processing platforms such as Storm or Apache Spark. A temporal latent space model for link prediction in dynamic social networks is proposed in [16]. This model assumes that each user lies in an unobserved latent space and interactions are more likely to occur between similar users in such a space. In addition, the model allows each user to gradually move position within the latent space as the network structure evolves over time. However, its validation is based on the assumption that the network evolves smoothly. In fact, some events may imply significant changes. Furthermore, and link weights are also neglected in this model.

Currently, the numbers of nodes in some social platforms, such as Facebook and WeChat, have reached the level of hundreds of millions, which need high efficient processing platforms, such as using the Hadoop ecosystem. In [17, 18] local similarity indices such as CN, AA and RA are computed within the MapReduce framework. However, MapReduce only provides two primitives, namely "Map" and "Reduce", which are not as efficient as Pregel, a professional graph computing framework [19]. In addition, due to the large number of input/output operations required in the MapReduce framework, the computational efficiency is much lower when compared to other memory-based big data processing platform (e.g., Apache Spark or Flink).

Yin et al.[20] propose a scalable approach for making inference about latent spaces of networks. A bag of triangular motifs is used as a succinct representation of networks in this approach. Based on this model, the link prediction method parsimonious triangular model (PTM) is competitive, however, this method is not suitable for distributed clusters. In [21], the authors introduce a dynamic mixed membership stochastic block model (DMMSB) to allow a linear Gaussian trend in the model parameters. However, DMMSB does not take the frequency of link into account. Yang et al. [22] introduce a new Nonnegative Matrix Factorisation (NMF) clustering method, Nonnegative Matrix Factorisation using graph Random walk (NMFR), which replaces the approximated matrix with its smoothed version using random walk. However, their approach does not scale well due to the high computation cost in each iteration. In [23], a new approach named HottTopixx (Hott), based on linear programming, is introduced, which aims to compute nonnegative matrix factorisations (NMFs). Unfortunately, such approach is designed to factorise the traditional rectangle matrices, and scalable approaches in symmetric graph factorisation are much less investigated than rectangle matrix factorisation such as Hott.

3. Main Definitions and Criteria

As discussed above, the dynamical properties of a network play an important role in predicting its topology. The temporal link prediction algorithm focuses on the dynamics of the network at time $t = 1, \dots n$ and it is defined as:

$$G = \{g_1, \dots, g_n\}, \tag{2}$$

where g_t represents the snapshot of the network at time t. Therefore, the temporal link prediction can be expressed as:

$$G \to g_{n+1}, \tag{3}$$

where G contains a network evolution sequence, and g_{n+1} is the prediction result of the network topology at the subsequent time interval. The process of

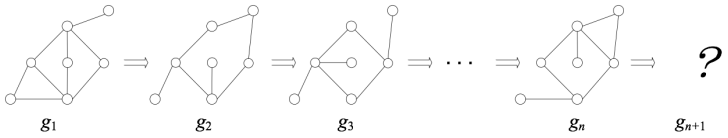

Figure 1: The process of the temporal link prediction.

the temporal link prediction is shown in Figure 1.

In this article, we use *Precision* and AUC (Area Under the receiver operating characteristic Curve) as the evaluation metrics for link prediction. The former is a direct evaluation of the accuracy of an algorithm, defined as the proportion of the correctly identified predicted links. In other words, if m is the number of correctly identified predicted links and l is their overall number, then

$$Precision = \frac{m}{l}. \tag{4}$$

AUC is calculated by randomly selecting one link from the test set and comparing it with the randomly selected non-existent link. First, suppose there are two nodes, x and y, which are connected at the next time window. Then, randomly select two other nodes x' and y', which are not connected during the same time window. Calculate S_{xy}^{WCN} and $S_{x'y'}^{WCN}$, over n iteration. Let n' and n'' be the number of times such that $S_{xy}^{WCN} > S_{x'y'}^{WCN}$ and $S_{xy}^{WCN} = S_{x'y'}^{WCN}$, respectively. Then, AUC is defined as:

$$AUC = \frac{n' + 0.5n''}{n}. \tag{5}$$

For more details on AUC, refer to [1]. If all scores are randomly generated, $AUC = 0.5$. The closer AUC is to 1, the better the prediction is.

4. Description of the Algorithm

4.1. Compression of Time-series Networks

The temporal information related to complex networks is largely defined by the dynamical properties of the interactions between nodes, which often reflect

changes of the relationships between the corresponding nodes. For example, a gradual reduction of communications between two individuals in a social network might indicate that the corresponding relationship is weakening.

The first step of DTLPLP focuses on embedding the historical network snapshots, $\{g_1, \ldots, g_n\}$, into a weighted network with related temporal information. We define τ to be the time window length, and the network snapshots during the previous τ time intervals are considered as the training set. For two nodes x and y, if the interaction times of the previous τ time intervals is $\{C_1, \ldots, C_\tau\}$, then the weight of the link (x, y) is defined as:

$$w_{x,y} = \sum_{i=1}^{\tau-1} (C_{i+1} - C_i)\delta^{\tau-i} + C_\tau, \tag{6}$$

where $\delta \in [0, 1]$ is the decay factor, which represents the influence of historical information. As a consequence, greater values of δ imply a more significant influence of historical evolution on the prediction results. Furthermore, if the interaction frequency (in terms of the number of messages between two nodes over a period of time) drops significantly, then $w_{x,y}$ may be less than 0, and it may be necessary to remove the corresponding links.

4.2. Link Prediction Based on Label Propagation

By considering the snapshots of the network within a specific time window, its dynamical properties can be effectively investigated. Subsequently, node labels are initialised, which are propagated to adjacent nodes. As opposed to traditional label propagation algorithms, DTLPLP enables the label attributes to be extended. In fact, in addition to the unique identifier ID of each node, each label also has an associate similarity value. In particular, during the propagation process the labels are propagated to their adjacent nodes, and they are updated based on the weights of the incident links, whilst their similarity values are aggregated by ID, and the link scores between nodes are generated. The detailed workflow of DTLPLP is as follows:

Step 1: Initialisation of labels. Each node is initialised as $(ID, 1)$.

Step 2: First iteration. All labels are propagated to their neighbouring nodes. During the process, the value of each label is updated via

$$v = v \cdot w, \tag{7}$$

where v is the similarity value, and w is the weight of the corresponding link. After the first iteration, the new set of labels will replace the initialised ones.

Step 3: Second iteration. The issues concerning self-similarity and backtracking of labels should be prevented. In other words, the case when the label propagated from node A to node B in the first iteration is propagated back to A in the second iteration must be avoided. This is achieved via the following steps:

1. Define an *empty* set L for each link in the network.
2. Determine whether the *key* of each label in the source node is equal to the ID of destination node. If so, the label will not be propagated via this link. Otherwise, add this label to L.
3. Update the similarity values for all labels in L. Recall that the weights of the links (x, y) and (y, z) are $w_{x,y}$, and $w_{y,z}$, respectively. The *similarity contribution* of z to x and y is defined as:

$$SC_{z\to(x,y)} = (w_{x,y} \cdot w_{y,z})^{\alpha}, \tag{8}$$

where α is the weight influence factor, and the higher its value is, the greater the weight influence will be.

4. Finally, merge L with the label set of the destination node.

Step 4: similarity aggregation. For each node, the similarity values of the labels are aggregated by ID, and the link scores between nodes are subsequently generated.

The contribution of the similarity of one-hop and two-hop neighbour nodes is fully considered by DTLPLP. The contribution of the similarity value of a one-hop neighbour is used as the weight of the link between these adjacent nodes.

The similarity contribution value of the two-hop neighbour is calculated with (8).

4.3. Distribution and Parallelisation

As opposed to the matrix calculation approach adopted by most existing algorithms, the iterative process of DTLPLP is natively distributed and can be parallelised based on the Bulk Synchronous Parallel (BSP) model. Firstly, the network structure is decomposed into a set of triplets, which are connected through the existing links according to a one-to-one correspondence. Each triplet contains a source node, a destination node and the link between them. The network structure can be subsequently distributed/stored on multiple computing nodes and the calculations spread across the local triples in each node. The distributed parallelisation of the algorithm is achieved by dividing the label propagation process for the whole network into the sum of the calculation processes of each triplet, according to the "divide and conquer" paradigm. In our network decomposition method, only one copy is stored for each link. Given that multiple links in different computing nodes share the same node, some of them may have both the original and multiple copies, after each iteration. Thus, synchronisation between the different nodes is necessary to ensure the accuracy of the next iteration calculation.

The iterative process of label propagation is carried out by computing the triplets. Each iteration is divided into three steps: node calculation, label propagation and information synchronisation. More specifically, the node calculation is responsible for grouping received labels into an array at each node, and label propagation transfers labels with similarity information to its neighbours. Finally, during the information synchronisation step, for all the copies associated with the same node communicating with their original node, their own set of labels are merged with the original node, whose set of labels are synchronised across all its copies. After the information synchronisation step is completed, the next iteration round is initiated.

More specifically, the proposed temporal link prediction algorithm is distributed and parallelised according to the following steps:

1. The input includes: time window length, prediction time, weight correction factor and the set of links with temporal information.
2. Based on the time window length τ and the prediction time T, links within the time interval $[T-\tau, T)$ are identified
3. The number of interactions between nodes, and temporal weight of links are calculated via (4);
4. The temporal-information of a network is defined from a set of links with temporal-weight and a label is subsequently generated, which includes the node ID and the initial similarity value at each node.
5. The temporal information of the network in a distributed environment is iteratively computed.
6. The similarity value of labels in each node is aggregated and links are assessed. Each iteration in this step follows the BSP model and is at the core of DTLPLP.
7. Finally, the output of the algorithm is the link score between nodes.

Each iteration can be divided into three actions: *node calculation*, *sending messages* (or labels) and *information synchronisation*. More specifically,

- The node calculation terminates after two iterations, and during the first one, an empty label set is sent to all nodes as the starting parameter for the first calculation action.
- The first two actions are designed to be performed on a single triplet, which are the basic computation units in the parallelised DTLPLP, whose pseudo-code are described in Algorithms 1 and 2, respectively.
- The synchronisation action is automatically completed by the network storage system.

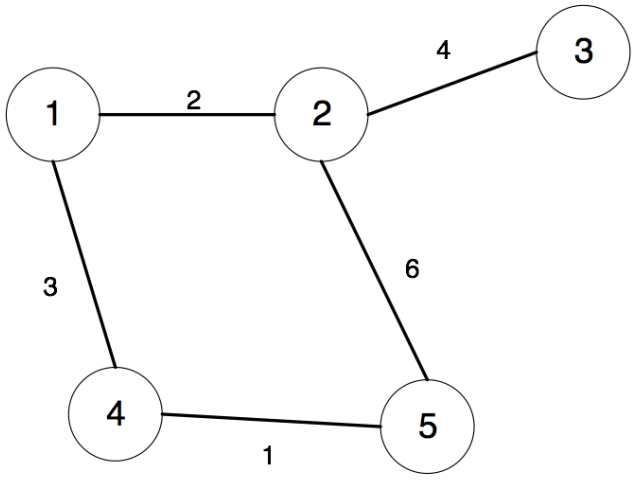

Figure 2: An example of weighted network with temporal information on links. Suppose that we have already known the weight fitted by temporal information between nodes.

Consider the network shown in Figure 2 as an example, where $\alpha = 0.2$ and the procedure of label propagation is shown in Table 1.

Figure 3 shows the topological structure of the network after two iterations with the top 6 link scores as the predicted links.

5. Evaluation

5.1. Experimental Platform

As discussed above, we have implemented the proposed link prediction algorithm using the Pregel interface provided by the Spark GraphX, which is designed from the Bulk Synchronous Parallel (BSP) model introduced by Google [19] . In particular, the input is a directed network, and each node has a unique node identifier, which contains a modifiable attribute. The directed links are defined via their source and destination nodes, which also have modifiable user-defined attributes. A typical calculation procedure in Pregel consists of the following steps: firstly, it reads the input network, which is initialised. A series of steps is subsequently executed until the end of the complete computation,

Table 1: Iterations of the process of the example depicted in Figure 2.

	Initialisation	**First Iteration**	**Second Iteration**	**Aggregate Messages**
Node 1	$\{(1,1)\}$	$\{(2,2),$ $(4,3)\}$	$\{(1,2),(3,4),$ $(3,1.52),(5,1.64),$ $(5,1.25)\}$	$\{(2,2),(4,3)$ $(2,1.52),$ $(5,2.89)\}$
Node 2	$\{(2,1)\}$	$\{(1,2),$ $(3,4)$ $(5,6)\}$	$\{(1,2),(3,4),$ $(5,6),(4,1.43),$ $(4,1.43)\}$	$\{(1,2),(3,4)$ $(5,6),(4,2.86)\}$
Node 3	$\{(3,1)\}$	$\{(2,4)\},$	$\{(2,4),$ $(1,1.52),(5,1.89)\}$	$\{(2,4),(1,1.52),$ $(5,1.89)\}$
Node 4	$\{(4,1)\}$	$\{(1,3),$ $\{(5,1)\}$	$\{(1,3),(5,1),$ $(2,1.43),(2,1.43)\}$	$\{(1,3),(5,1),$ $(2,1.86)\}$
Node 5	$\{(5,1)\}$	$\{(2,6),$ $(4,1)\}$	$\{(2,6),(4,1),$ $(1,1.25),(1,1.64),$ $(3,1.89)\}$	$\{(2,6),(4,1),$ $(1,2.89),$ $(3,1.89)\}$

which are separated by some global synchronisation, and generating the relevant results before its termination. In GraphX which is one of the most important component of Spark, an open source implementation of Pregel is provided. As opposed to the standard Pregel, in GraphX messages are aggregated before being sent to the same node. Each super step of the GraphX Pregel performs three actions:

Message sending: each node sends its own attribute to their neighbour nodes according to the defined rules. These rules include transmission direction (transmission along the outgoing side, transmission along the incoming side or bidirectional transmission), and transmission conditions (the attribute meets the conditions are sent or not sent).

Message aggregation: as discussed above, GraphX Pregel aggregates the messages sent to the same node, which is subsequently sent to the destination

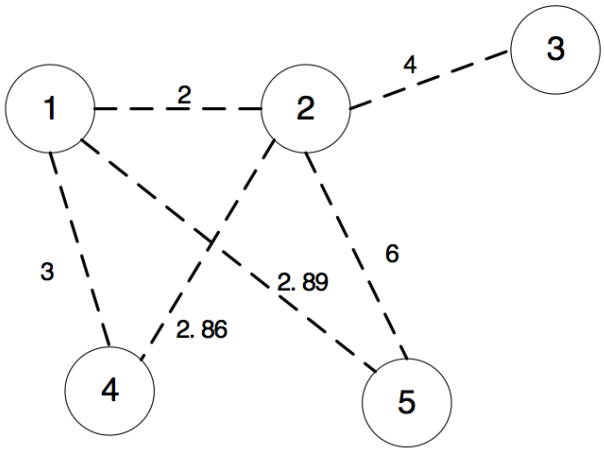

Figure 3: Link prediction result.

nodes.

Node calculation: after a node receives the aggregated message from its neighbours, its attribute is updated based on the received aggregation.

As the network storage of GraphX is based on a node segmentation approach, in a distributed environment each node may have one original and multiple copies. After each super step, a synchronisation across all the original and copies is required to ensure that the information is consistent. This step is handled by Spark.

5.2. Description of Datasets

The validation of the proposed method is based on the Enron e-mail [24], as well as the General Relativity and Quantum Cosmology Scientific Collaboration networks [25]. The former contains 252,759 e-mails, including the those occurred between the 151 Enron Corporation executives within December 1999 and September 2000. The communication networks before each month was taken as training set, where each node represents an e-mail address linked by the corresponding communication divided into months, and the weight depends

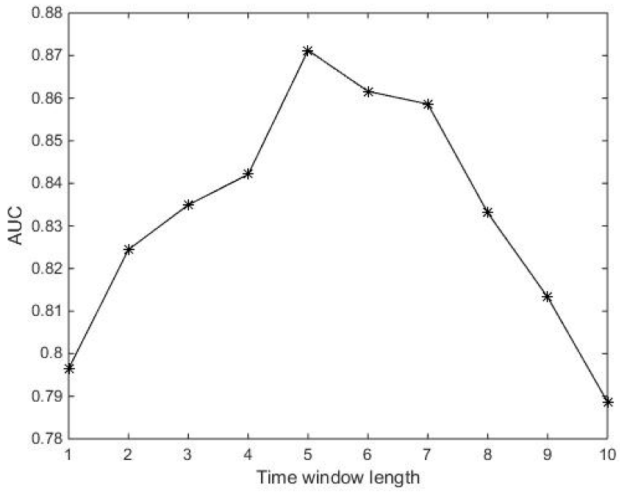

Figure 4: Effect of time window on the prediction.

on the number of exchanged emails.

The General Relativity and Quantum Cosmology Scientific Collaboration network is based on the collaborative work among scientists in this field between January 1993 and April 2003. The network has 5242 nodes and 14496 links, but they are not associated with any temporal information. In order to test the scalability of the DTLPLP, each link is given a random weight between 1 and 10, and subsequently parallel acceleration test based on Pregel was performed.

5.3. Experimental Results

5.3.1. The Effect of Time Window on Prediction

The first validation component focused on assessing the effect of the time window length τ on the prediction performance of the algorithm. Specifically, we considered a weight correction factor $\alpha = 0.6$ and a temporal information influence factor $\sigma = 0.8$. The average AUC for the first five months in the test is shown in Figure 4, which demonstrates that the algorithm achieves the best prediction when the time window length is approximately 6.

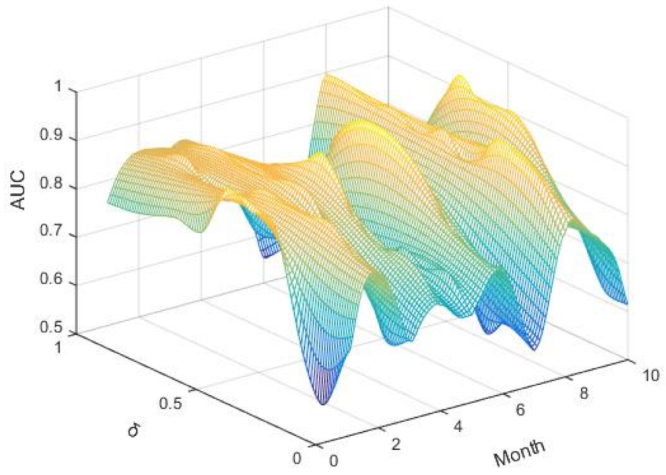

Figure 5: Effect of σ on the performance of the algorithm.

The *AUC* of the algorithm is 8% higher than that of the prediction result without being updated by the temporal information. In particular, a short time window length suggests a weaker influence of the dynamics of the network on the prediction performance. In contrast, a longer time window length implies that historical information will have a negative impact on the accuracy of the algorithm. Therefore, for the temporal information method described by (4), the optimal time window length appears to be between 5 and 7.

5.3.2. Influence of Historical Influence Factor σ

Recall that σ represents the influence of temporal information, whose value has direct impact on link prediction. In the investigation of the effect of σ, we assumed to have a time window of length 5 and $\alpha = 0.2$, as depicted in Figure 5. This also shows that the use of temporal information in the algorithm can improve the effect of the link prediction, and for $\sigma \approx 0.5$ the prediction results tend to be stable. If $\sigma \approx 1$, then the best prediction results can be potentially obtained, but their stability is decreased. Considering the prediction effect and the stability of the algorithm comprehensively, the appropriate value of σ appears to be approximately 0.5.

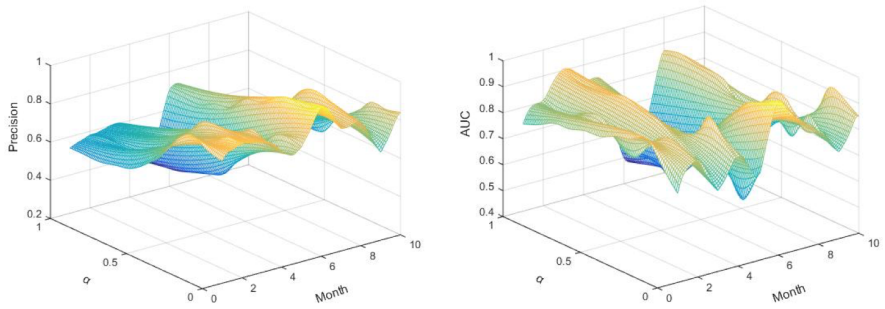

Figure 6: *Precision* and *AUC* of DTLPLP.

5.3.3. Influence of Weight Correction Factor α

Based on the above experimental results, we assumed $\sigma = 0.5$ with a time window of length 5. The top 50 values of the prediction link scores were considered when evaluating the precision of the proposed method, which account for 30% of the total links of each month. The overall precision and the *AUC* of the algorithm are shown in Figures 6(a) and 6(b). It can be observed that when the weight correction coefficient α is close to 1, the fluctuation of prediction effect is bigger, and the best and worst results appear in this region. When $\alpha \approx 0$, the prediction effect of the algorithm tends to be stable. As a consequence, the appropriate value of α is in the range between 0.2 and 0.3.

5.3.4. Comparison with Other Algorithms

We also implemented the CN and RA algorithms which have a better performance in conventional link prediction tasks. The WCN algorithm was also tested on the Enron e-mail network, with $\alpha = 0.8$, where it achieved the best prediction effect. Figures 7(a) and 7(b) depict the comparison between DTLPLP, CN, RA and WCN for $\alpha = 0.2$, which demonstrate that DTLPLP has a better performance in terms of accuracy (*precision* and *AUC*).

Furthermore, Figure 4 suggests that integrating weight information with the

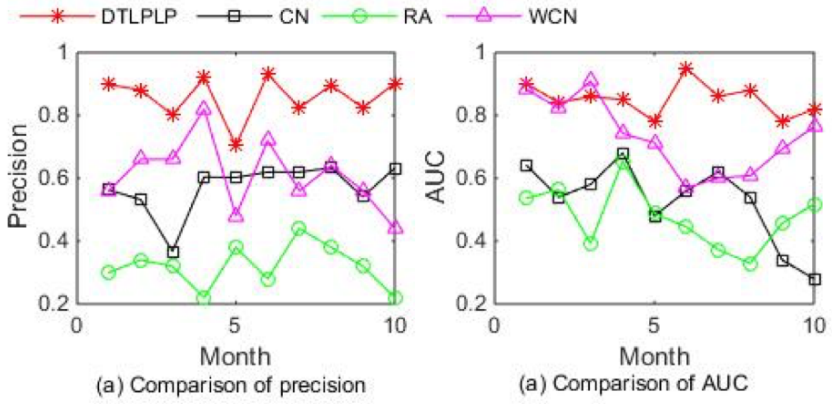

Figure 7: Comparison of DTLPLP with other similar algorithms as discussed in Section 5.3.4.

link prediction algorithm improve its performance.

We implemented DTLPLP on other datasets, including Infectious, Facebook (WOSN) and HepPh [26], to carry out a comparison with BCGD [16], PTM [20], DMMSB [21], NMFR [22] and Hott [23]. In [16], the prediction performance of BCGDG (global BCGD) is demonstrated to be more accurate than BCGDL (local BCGD) and BCGDI (incremental BCGD). Therefore, BCGDG is selected for evaluation purposes.

As depicted in Figure 9, on both datasets of Infectious and HepPh, the prediction performance of DTLPLP is superior to other algorithms. However, on the Facebook dataset (WOSN), the performance of DTLPLP is weaker than BCGDG due to the fact that most links in the next "snapshots" refer to new connections, which had never appeared in the past. This further suggests that DTLPLP is better at predicting dynamic interaction rather than existing social relations. However, it is also clear that DTLPLP capability of predicting new links is still superior to most current algorithms.

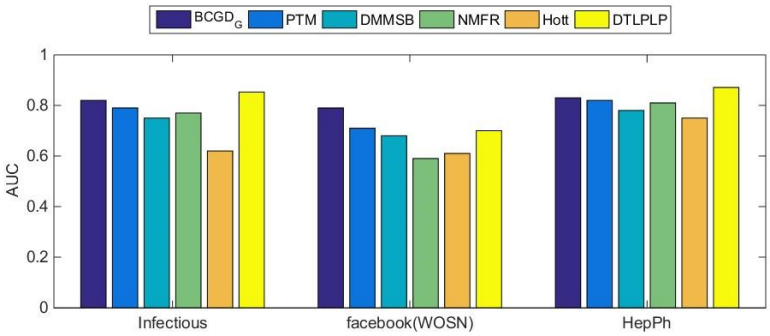

Figure 8: Further comparison of DTLPLP with other advanced Algorithms.

Table 2: Hardware configuration of the test cluster.

Cluster Nodes	**Memory(GB)**	**Number of CPU Cores**
Master	10	8
Slave 1	20	8
Slave 2	20	8
Slave 3	20	8

5.3.5. Accelerated Testing in Distributed Environment of DTLPLP

This section focuses on the evaluation of the algorithm in a distributed environment, consisting of a computer cluster with four machines (one master node and three slave nodes). Specific hardware configuration is shown in Table 2.

The network bandwidth between nodes is in the range between $7.2MB/s$ and $8.1MB/s$, with Spark 1.6.1, Hadoop is 2.4.1, and YARN as the resource manager. A total of six executors were initiated during the execution process, with 5GB memory allocated for each of them.

In order to increase the degree of parallelism, 3, 18, and 30 partitions were considered. For each different partition, the data was evaluated three times, and Figure 9 depicts the average running time as an efficiency indicator.

In Spark, each CPU core can only calculate one partition at a time. Therefore, the number of cores that the resource manager assigns to Spark determines

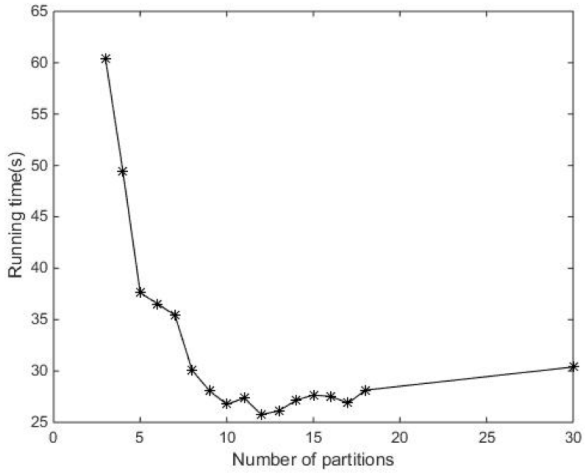

Figure 9: Parallel running time of DTLPLP.

the corresponding number of partitions, which was 12 due to limitations of the hardware configuration. When the number of partitions is less than the maximum degree of parallelism, the running time of DTLPLP decreased as number of partitions increased. However, when that limit is reached, CPU utilisation attains saturation. Furthermore, if the number of partitions is increased, running time will slowly increase. This is due to the fact that a bigger number of partitions will not result in an increase of the degree of parallelism, and switching partitions and network communication will add further computational time.

6. Conclusion

Link prediction has been successfully applied in many contexts, and it has been shown to have a number of applications in complex network research. However, current link prediction algorithms do not take into consideration the underlying network dynamics and cannot efficiently scale-up to large organisations.

In this article, we have proposed DTLPLP, a distributed temporal link prediction algorithm based on label propagation, which is particularly suitable for a

distributed environment. More specifically, the DTLPLP algorithm is implemented in a distributed network analysis framework, namely Pregel, and it can be readily applied to large-scale, real-world network datasets. Experiments have been performed by using the Spark framework, demonstrating a superior performance compared to similar link prediction algorithms.

Furthermore, complex networks have specific dynamical properties, which often exhibit an unstable behaviour when external events occur. Therefore, efficient link prediction algorithms need to investigate information fluctuations embedded in the network. Future work will address a full investigation of the perturbation effect on the underlying network, whilst optimising the overall stability of the system.

References

[1] L. Linyuan, T. Zhou, Link prediction in complex networks: A survey, ACM Comput. Surv. (69:1-69:33).

[2] D. Liben-Nowell, J. M. Kleinberg, The link-prediction problem for social networks, JASIST 58 (2007) 1019–1031.

[3] G. Carullo, A. Castiglione, A. De Santis, F. Palmieri, A triadic closure and homophily-based recommendation system for online social networks, World Wide Web 18 (6) (2015) 1579–1601. doi:10.1007/s11280-015-0333-5.

[4] A. Anil, D. Kumar, S. Sharma, R. Singha, R. Sarmah, N. Bhattacharya, D. Kumar, S. Sharma, R. Singha, R. Sarmah, N. Bhattacharya, S. R. Singh, Link prediction using social network analysis over heterogeneous terrorist network, in: IEEE International Conference on Smart City/socialcom/sustaincom, 2015, pp. 267–272.

[5] Y. Sakata, K. Eguchi, Cross-lingual link prediction using multimodal relational topic models, in: 5th IEEE/ACIS International Conference on Com-

puter and Information Science, ICIS 2016, Okayama, Japan, June 26-29, 2016, 2016.

[6] C. Ahmed, A. ElKorany, Enhancing link prediction in twitter using semantic user attributes, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 201, ACM, 2015, pp. 1155–1161.

[7] B. Kaya, M. Poyraz, Finding relations between diseases by age-series based supervised link prediction, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks, 2015, pp. 1097–1103.

[8] P. Minervini, C. d'Amato, N. Fanizzi, Efficient energy-based embedding models for link prediction in knowledge graphs, J. Intell. Inf. Syst. (2016) 91–109.

[9] T. Zhou, L. Lu, Y. Zhang, Predicting missing links via local information, The European Physical Journal B-Condensed Matter and Complex Systems (2009) 623–630.

[10] L. Lu, T. Zhou, Link prediction in weighted networks: The role of weak ties, EPL (Europhysics Letters).

[11] L. Gao, J. Yang, G. Qin, Methods for pattern mining in dynamic networks and applications, J. Journal of Software (2013) 2042–2061.

[12] Z. Denga, S. Lao, L. Bai, A temporal link prediction method based on link prediction error correction, J. journal of Electronics and Information Technology (2014) 325–331.

[13] Z. Zhao, Y. Jia, Y. Wang, X. Jin, X. Cheng, Temporal link prediction based on dynamic heterogeneous information network, J. Computer Research and Development (2015) 1735–1741.

[14] N. M. A. Ibrahim, L. Chen, Link prediction in dynamic social networks by integrating different types of information, J. Applied Intelligence 42 (4) (2015) 738–750.

[15] P. Zhao, C. Aggarwal, G. He, Link prediction in graph streams, in: IEEE International Conference on Data Engineering, 2016, pp. 553–564.

[16] L. Zhu, D. Guo, J. Yin, G. V. Steeg, A. Galstyan, Scalable temporal latent space inference for link prediction in dynamic social networks, IEEE Trans. Knowledge and Data Engineering 28 (10) (2016) 2765–2777.

[17] J. Rao, B. Wu, Y. Dong, Parallel link prediction in complex network using mapreduce, J. Journal of Software 23 (12) (2012) 3175–3186.

[18] H. Yuan, Y. Ma, F. Zhang, M. Liu, A distributed link prediction algorithm based on clustering in dynamic social networks, in: IEEE International Conference on Systems, Man, and Cybernetics, 2015, pp. 1341–1345.

[19] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, G. Czajkowski, Pregel: a system for large-scale graph processing, in: ACM Symposium on Parallelism in Algorithms and Architectures, 2009, pp. 135–146.

[20] J. Yin, Q. Ho, E. P. Xing, A scalable approach to probabilistic latent space inference of large-scale networks, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13, Curran Associates Inc., USA, 2013, pp. 422–430.
URL http://dl.acm.org/citation.cfm?id=2999611.2999659

[21] W. Fu, L. Song, E. P. Xing, Dynamic mixed membership blockmodel for evolving networks, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, pp. 329–336. doi:10.1145/1553374.1553416.
URL http://doi.acm.org/10.1145/1553374.1553416

[22] Z. Yang, T. Hao, O. Dikmen, X. Chen, E. Oja, Clustering by nonnegative matrix factorization using graph random walk, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1079–1087.

URL http://papers.nips.cc/paper/4845-clustering-by-nonnegative-matrix-factorization-usi
pdf

[23] B. Recht, C. Re, J. Tropp, V. Bittorf, Factoring nonnegative matrices with linear programs, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1214–1222.
URL http://papers.nips.cc/paper/4518-factoring-nonnegative-matrices-with-linear-program
pdf

[24] J. I. Adibi, Enron email dataset.

[25] General relativity and quantum cosmology collaboration network.

[26] J. Kunegis, Konect: The koblenz network collection, in: Proceedings of the 22Nd International Conference on World Wide Web, WWW '13 Companion, ACM, New York, NY, USA, 2013, pp. 1343–1350. doi:10.1145/2487788.2488173.
URL http://doi.acm.org/10.1145/2487788.2488173

Algorithm 1 Sending of Message

1: **Input:** the source node $srcId$ with attribute $srcAttr$, and destination node $dstId$ with attribute $dstAttr$
2: **Output:** the message $srcArr$ sent to the source node $srcId$ and the message $dstArr$ sent to the destination node $dstId$.
3: Initialise the sets $srcArr$ and $dstArr$
4: **for** $dstElem \leftarrow dstAttr$ **do**
5: **if** $dstElem.key == dstId$ **then**
6: $dstElem.key, dstElem.value * attr) \rightarrow srcArr$
7: **else**
8: **if** $srcElem.key! = srcId$ **then**
9: $dstElem.key, (dstElem.value * attr)^{\alpha}) \rightarrow srcArr$
10: **end if**
11: **else**
12: $NULL \rightarrow srcArr$
13: **end if**
14: **end for**
15: Filter out the null values in $srcArr$
16: **for** $srcElem \leftarrow srcAttr$ **do**
17: **if** $srcElem.key == srcId$ **then**
18: $srcElem.key, srcElem.value * attr) \rightarrow dstArr$
19: **else**
20: **if** $srcElem.key! = srcId$ **then**
21: $srcElem.key, (srcElem.value * attr)^{\alpha} \rightarrow dstArr$
22: **end if**
23: **else**
24: $NULL \rightarrow dstArr$
25: **end if**
26: **end for**
27: Filter out the null values in $dstArr$
28: **return** $(dstId, dstArr), (srcId, srcArr)$

Algorithm 2 Node Calculation

1: **Input:** id of node Id, attribute $vdata$ of node, the received message msg
2: **Output:** update value of node
3: **if** $vdata.length == 1$ **and** $vdata[0].key == Id$ **and** $message.length > 0$ **then**
4: **return** msg
5: **else**
6: **return** $vdata + msg$
7: **end if**